How to overcome the numerical instability of the scheme of divided differences?

Alicja Smoktunowicz, Przemysław Kosowski and Iwona Wróbel

Faculty of Mathematics and Information Science, Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw, Poland

Abstract

The scheme of divided differences is widely used in many approximation and interpolation problems. Computing the Newton coefficients of the interpolating polynomial is the first step of the Björck and Pereyra algorithm for solving Vandermonde systems of equations (Cf. [1]). Very often this algorithm produces very accurate solution. The problem of determining the Newton coefficients is intimately related with the problem of evaluation the Lagrange interpolating polynomial, which can be realized by many algorithms. For these reasons we use the uniform approach and analyze also Aitken's algorithm of the evaluation of an interpolating polynomial. We propose new algorithms that are always numerically stable with respect to perturbation in the function values and more accurate than the Aitken's algorithm and the scheme of divided differences, even for complex data.

AMS subject classification: 65D05, 65G50

Keywords: Numerical stability, condition number, Lagrange form, Newton form, divided differences, Aitken algorithm

1 Introduction

The aim of this paper is to study the numerical stability of algorithms of evaluating the Lagrange interpolating polynomial $w_N(z) = \sum_{j=0}^{N} f_j \prod_{i=0, i\neq j}^{N} \frac{z-z_i}{z_j-z_i}$ and computing the Newton coefficients $c_0, \ldots, c_N$ of $w_N(z)$. We assume that $z_0, z_1, \ldots, z_N$ are pairwise distinct data points and $f_0, f_1, \ldots, f_N$ are associated values (real or complex) of function f. Then $w_N(z)$ interpolates function $f(z)$ at points $z_0, z_1, \ldots, z_N$, i.e. $f(z_j) = f_j = w_N(z_j)$ for $j = 0, 1, \ldots, N$ and $w_N(z) = \sum_{j=0}^{N} c_j \prod_{i=0}^{j-1} (z - z_i)$.

The problem of determining the Newton coefficients is intimately related with the problem of evaluation the Lagrange interpolating polynomial, which can be realized by many algorithms. For these reasons we use the uniform approach and consider the following general problem.

Problem I. For given $t \in \mathbb{C}$, $z_0, z_1, \ldots, z_N \in \mathbb{C}$ (distinct) and $f_0, f_1, \ldots, f_N$ compute

$$p_n(z;t) = \sum_{j=0}^{n} f_j \prod_{i=0,\, i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad n = 0, 1, \ldots, N. \tag{1}$$

In the special case of $z = tz_j$ for some j we obtain $p_n(z,t) = f_j\, t^n$.

Notice that for $t = 1$ we have

$$p_n(z;1) = w_n(z) = \sum_{j=0}^{n} f_j \prod_{i=0,\, i\neq j}^{n} \frac{z - z_i}{z_j - z_i}, \qquad n = 0, 1, \ldots, N \tag{2}$$

and for $t = 0$, $z = 1$ we get

$$p_n(1;0) = c_n = \sum_{j=0}^{n} f_j \prod_{i=0,\, i\neq j}^{n} \frac{1}{z_j - z_i}, \qquad n = 0, 1, \ldots, N. \tag{3}$$

Usually the Newton coefficients are computed by the scheme of divided differences. However, it is known (Cf. [6]) that the reputation of this algorithm is not so good. Its numerical properties depend on the distribution and ordering of the interpolation points. A. Kiełbasiński ([4], pp. 51-53) proved that if the nodes are real and monotonically ordered, i.e. $z_0 < z_1 < \ldots < z_N$ or $z_0 > z_1 > \ldots > z_N$, then every Newton coefficient computed in fl is the exact divided difference for slightly perturbed values f_j (see Proposition 1), that is numerically stable in the following sense.

Definition 1 *We say that an algorithm W of computing*

$$p_n(z;t) = \sum_{j=0}^{n} f_j \prod_{i=0,\ i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad n = 0, 1, \ldots, N$$

is numerically stable (with respect to the data $f_0, f_1, \ldots, f_N$) if the values $\tilde{p}_n(z;t)$ computed by W in floating point arithmetic satisfy

$$\tilde{p}_n(z;t) = \sum_{j=0}^{n} [f_j(1 + \delta_j^{(n)})] \prod_{i=0,\ i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad |\delta_j^{(n)}| \leqslant \epsilon_M k_N \tag{4}$$

and k_N is a modest constant.

It is easy to check that (4) is equivalent to

$$|\tilde{p}_n(z;t) - p_n(z;t)| \leqslant \epsilon_M k_N \sum_{j=0}^{n} |f_j| \prod_{i=0,\ i\neq j}^{n} \frac{|z - tz_i|}{|z_j - z_i|}, \qquad n = 0, \ldots, N. \tag{5}$$

For all set of values we have

$$\frac{\max\limits_{n=0,\ldots,N} |\tilde{p}_n(z;t) - p_n(z;t)|}{\max\limits_{n=0,\ldots,N} |p_n(z;t)|} \leqslant \epsilon_M k_N cond_f(z_0, \ldots, z_N; z; t), \tag{6}$$

where

$$cond_f(z_0, \ldots, z_N; z; t) = \frac{\max\limits_{n=0,\ldots,N} \sum\limits_{j=0}^{n} |f_j| \prod\limits_{i=0,\ i\neq j}^{n} \frac{|z - tz_i|}{|z_j - z_i|}}{\max\limits_{n=0,\ldots,N} |p_n(z;t)|} \tag{7}$$

is the measure of sensitivity of Problem I with respect to the data $f_0, f_1, \ldots, f_N$ and will be referred to as the condition number.

In Section 1 we review two standard methods, i.e. Aitken's interpolation algorithm and the scheme of divided differences. Section 3 presents a new efficient algorithm for computing (1). Error analysis of these algorithms is given in Section 4. The new algorithm is shown to be numerically stable in sense (4), even for complex data, opposite to the scheme of divided differences and Aitken's algorithm. We also show that these two algorithms are numerically stable if the knots are real and monotonically ordered (Cf. [4], pp. 51-53, [9]). Numerical tests in *MATLAB* included in Section 5 confirm theoretical results.

The new algorithm doesn't require any special ordering of interpolation points, so we can try to reorder the knots to achieve small relative error (see (6)). However, it seems to be a hard problem to minimize a condition number of Problem I (see (7)). It was observed that the Leja ordering (Cf. [2], [8]) may be a good choice. To reorder the points $z_0, z_1, \ldots, z_N$ by Leja, first choose z_0 satisfying $|z_0| = \max_j |z_j|$, and then determine z_j, $j = 1, 2, \ldots, N-1$ such that $\prod_{k=0}^{j-1} |z_j - z_k| = \max_{i \geqslant j} \prod_{k=0}^{j-1} |z_i - z_k|$.

Although in many cases helpful, sometimes it gives results comparable to or even worse than monotone ordering, even for equidistant points (see Section 5, Table 3).

2 Algorithms

The simplest way to compute Newton coefficients (3) is the well known scheme of divided differences, which can be visualized as follows:

$$c_j^{(0)} := f_j, \qquad j = 0, 1, \ldots, N$$

$$\begin{tikzcd}
z_0 & c_0^{(0)} \arrow[dr] & & & & & \\
& & c_0^{(1)} \arrow[dr] & & & & \\
z_1 & c_1^{(0)} \arrow[ur] \arrow[dr] & & c_0^{(2)} & & & \\
& & c_1^{(1)} \arrow[ur] & & & \arrow[dr] & \\
z_2 & c_2^{(0)} \arrow[ur] & & \cdots & \cdots & & c_0^{(N)} \\
& & \cdots & & & \arrow[ur] & \\
\cdots & \cdots & & c_{N-2}^{(2)} & & & \\
& & c_{N-1}^{(1)} \arrow[ur] & & & & \\
z_N & c_N^{(0)} \arrow[ur] & & & & &
\end{tikzcd}$$

where $c_j^{(n)}$ are the divided differences of n*th* order. They are defined in the following way

$$f(z_j, z_{j+1}, \dots, z_{j+n}) = c_j^{(n)} = \frac{c_j^{(n-1)} - c_{j+1}^{(n-1)}}{z_j - z_{j+n}}$$

for $j = 0, 1, \dots, N-n$, and $n = 1, 2, \dots, N$.

Computing the value of the Lagrange interpolating polynomial at a given point z (see formula (2)) can be realized by Aitken's algorithm (see [3], Problem 8 on p. 289).

The following Algorithm I can be used for computing both Newton coefficients and the value of interpolating polynomial, depending on z and t. For $t = 0$ and $z = 1$ algorithm reduces to the divided differences scheme, and for $t = 1$ it is Aitken's algorithm. For all other choices of z and t Algorithm 1 is a general way of evaluating (1).

Algorithm I

For given (real or complex) z_n and f_n, t and z this algorithm computes

$$p_n(z;t) = \sum_{j=0}^{n} f_j \prod_{i=0,\, i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad n = 0, 1, \dots, N.$$

```
Choose $t$ and $z$
$c_j^{(0)} := f_j, \quad j = 0, 1, \dots, N$
for  $n = 1, 2, \dots, N$
    for  $k = 0, 1, \dots, N-n$
        $c_k^{(n)} := \frac{(z - tz_{n+k})\, c_k^{(n-1)} - (z - tz_k)\, c_{k+1}^{(n-1)}}{z_k - z_{k+n}}$        (8)
    end
    $p_n(z;t) := c_0^{(n)}$
end
```

Then

$$c_k^{(n)} = \sum_{j=k}^{k+n} f_j \prod_{i=k,\, i\neq j}^{k+n} \frac{z - tz_i}{z_j - z_i}, \qquad n = 0, 1, \dots, N, \qquad k = 0, 1, \dots, N-n. \tag{9}$$

It means that for $t = 1$ $c_k^{(n)}$ is the Lagrange interpolating polynomial based on the knots $z_k, z_{k+1}, \ldots, z_{k+n}$ and for $z = 1$, $t = 0$ $c_k^{(n)}$ is the divided difference of n*th* order based on the knots $z_k, \ldots, z_{k+n}$.

The complexity of the divided differences and Aitken's algorithms is $\mathcal{O}(N^2/2)$ and $\mathcal{O}(3N^2/2)$ complex multiplications, respectively. The general algorithm requires $\mathcal{O}(5N^2/2)$ complex multiplications.

3 New algorithm

In this section we propose a new algorithm for computing (1). The idea is similar in spirit to the ones introduced in [9], [11] and [7], but our Algorithm II is more efficient and robust (see Theorem 2).

For $z \neq tz_j$, $j = 0, 1, \ldots, N$

$$\prod_{i=0,\ i\neq j}^{n} (z - tz_i) = \frac{A_n}{z - tz_j}, \qquad n = 0, 1, \ldots, N,$$

where

$$A_n = (z - tz_0)(z - tz_1) \cdot \ldots \cdot (z - tz_n). \tag{10}$$

Thus

$$p_n(z;t) = A_n \sum_{j=0}^{n} \frac{f_j}{(z - tz_j)\, w_j^{(n)}}, \qquad w_j^{(n)} = \prod_{i=0,\ i\neq j}^{n} (z_j - z_i). \tag{11}$$

Note that A_n has the following form

$$A_n = \begin{cases} (z - z_0) \cdot \ldots \cdot (z - z_n) & \text{for } t = 1, \\ 1 & \text{for } t = 0,\ z = 1. \end{cases}$$

We observe that

$$w_j^{(0)} = 1, \qquad w_j^{(n)} = w_j^{(n-1)}(z_j - z_n) \quad \text{for } j = 0, 1, \ldots, n-1,$$

$$A_n = (z - tz_n)A_{n-1}, \quad n = 0, 1, \ldots, N, \quad \text{with } A_{-1} = 1.$$

Then for $n = 0, 1, \ldots, N$ we have $p_n(z; t) = A_n \ B_n$ where

$$B_n = \sum_{j=0}^{n} b_j^{(n)}, \qquad b_j^{(n)} = \frac{f_j}{(z - tz_j)\, w_j^{(n)}}, \qquad j = 0, 1, \ldots, n. \tag{12}$$

We see that

$$b_j^{(0)} = \frac{f_j}{z - tz_j}, \qquad j = 0, 1, \ldots, N \tag{13}$$

and

$$b_j^{(n)} = \frac{b_j^{(0)}}{w_j^{(n)}}, \qquad j = 0, 1, \ldots, n. \tag{14}$$

Notice that if B_{n-1} is written in the form

$$B_{n-1} = b_0^{(n-1)} + b_1^{(n-1)} + \ldots + b_{n-1}^{(n-1)},$$

then we can rewrite B_n as follows

$$B_n = \frac{b_0^{(n-1)}}{(z_0 - z_n)} + \frac{b_1^{(n-1)}}{(z_1 - z_n)} + \ldots + \frac{b_{n-1}^{(n-1)}}{(z_{n-1} - z_n)} + \frac{b_n^{(0)}}{(z_n - z_0)(z_n - z_1)\ldots(z_n - z_{n-1})}.$$

Further, the formulae (11)-(12) let us write

$$b_j^{(n)} = \frac{b_j^{(n-1)}}{z_j - z_n}, \qquad j = 0, 1, \ldots, n-1, \quad n = 1, 2, \ldots, N \tag{15}$$

and

$$b_n^{(n)} = \frac{b_n^{(0)}}{\prod_{j=0}^{n-1} (z_n - z_j)}. \tag{16}$$

Algorithm II

For given (real or complex) z_n and f_n, t and $z \neq tz_j$ this algorithm computes

$$p_n(z; t) = \sum_{j=0}^{n} f_j \prod_{i=0,\ i \neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad n = 0, 1, \ldots, N.$$

Choose t and z
$b_j^{(0)} := \dfrac{f_j}{z - tz_j}, \qquad j = 0, 1, \ldots, N,$
$A_0 := z - tz_0$

```
$B_0 := b_0^{(0)}$
for  $n = 1, 2, \dots, N$
    $A_n := (z - t z_n) A_{n-1}$
    for  $j = 0, 1, \dots, n-1$
        $b_j^{(n)} := \dfrac{b_j^{(n-1)}}{z_j - z_n}$
    end
    $b_n^{(n)} := \dfrac{b_n^{(0)}}{\prod_{j=0}^{n-1}(z_n - z_j)}$
    $B_n := \sum_{j=0}^{n} b_j^{(n)}$
    $p_n(z;t) := A_n B_n$
end
```

The complexity of Algorithm II is $\mathcal{O}(N^2)$ flops.

Note that this algorithm can be implemented more efficiently than it's written now, namely $b_n^{(n)}$ can be computed along with $b_j^{(n)}$ in the inner loop for

$$b_n^{(n)} := \frac{(-1)^n \, b_n^{(0)}}{\prod_{j=0}^{n-1}(z_j - z_n)}.$$

We decided to write Algorithm II this way to make it more clear.

4 Error analysis of algorithms

We consider complex arithmetic (cfl) (Cf. [9]) implemented using real arithmetic (fl) with machine unit ϵ_M. We assume that for $x, y \in \mathbb{R}$ we have

$$fl(x \pm y) = (x \pm y)(1 + \delta), \qquad |\delta| \leqslant \epsilon_M. \tag{17}$$

Then

$$cfl(x \odot y) = (x \odot y)(1 + \delta), \qquad |\delta| \leqslant c_\odot \, \epsilon_M \quad \text{for } x, y \in \mathbb{C}, \tag{18}$$

where

$$c_\odot = \begin{cases} 1 & \text{for} \quad \odot \in \{+, -\}, \\ 1 + \sqrt{2} & \text{for} \quad \odot = *, \\ 4 + \sqrt{2} & \text{for} \quad \odot = /. \end{cases} \tag{19}$$

The constant $c_\odot$ is not significant, it depends on the implementation of floating point operations. In fact, the bounds of errors in Algorithms I and II are of the same form for real and complex arithmetic, only the constants are increased appropriately.

Our error analysis is uniform, it includes both real and complex cases, i.e. the data z_j, f_j, z and t can be either real or complex.

The values computed in fl or cfl will be marked with tilde in order to distinguish them from the exact ones.

4.1 Error analysis of Algorithm I

Theorem 1 *Assume that $z_j,\ f_j, \in \mathbb{C}$ for $j = 0, 1, \ldots, N$ and $z,\ t \in \mathbb{C}$ are exactly representable in cfl and*

$$cfl(tz_j) = tz_j, \qquad j = 0, 1, \ldots, N. \tag{20}$$

Then the values $\tilde{c}_k^{(n)}$ for $n = 0, 1, \ldots N,\ k = 0, 1, \ldots, N - n$ computed in cfl by Algorithm I satisfy

$$\tilde{c}_k^{(n)} = \sum_{j=k}^{k+n} [f_j(1 + \delta_{k,j}^{(n)})] \prod_{i=k,\, i\neq j}^{k+n} \frac{z - tz_i}{z_j - z_i}, \qquad |\delta_{k,j}^{(n)}| \leqslant \epsilon_M n\, d\, L^{n-1} + \mathcal{O}(\epsilon_M^2), \tag{21}$$

where

$$d = \begin{cases} 5 & \textit{in the real case,} \\ 8 + 2\sqrt{2} & \textit{in the complex case} \end{cases} \tag{22}$$

and

$$L = \max_{0\leqslant i<j<k\leqslant N} \frac{|z_i - z_j| + |z_j - z_k|}{|z_i - z_k|}. \tag{23}$$

Proof. By induction on n. It follows from (8) and (17)-(19) that for $n = 0, 1, \ldots N$, $k = 0, 1, \ldots, N - n$ the computed quantities satisfy

$$\tilde{c}_k^{(n)} := \frac{(1 + \alpha_k^{(n)})(z - tz_{n+k})\tilde{c}_k^{(n-1)} - (1 + \beta_k^{(n)})(z - tz_k)\tilde{c}_{k+1}^{(n-1)}}{z_k - z_{k+n}}, \tag{24}$$

where

$$|\alpha_k^{(n)}|,\ |\beta_k^{(n)}| \leqslant \epsilon_M\, d + \mathcal{O}(\epsilon_M^2). \tag{25}$$

It is clear that for $n = 1$ the result is immediate, since

$$\tilde{c}_k^{(1)} = f_k\,(1+\alpha_k^{(1)})\frac{z - tz_{k+1}}{z_k - z_{k+1}} + f_{k+1}\,(1+\beta_k^{(1)})\frac{z - tz_k}{z_{k+1} - z_k},$$

so we can take $\delta_{k,k}^{(1)} = \alpha_k^{(1)}$ and $\delta_{k,k+1}^{(1)} = \beta_k^{(1)}$, $k = 0, 1, \ldots, N-1$.

Now assume that (21) is true for $n-1$. We will prove that it holds also for n. By assumption for $n-1$ we get

$$(z - tz_{n+k})\,\tilde{c}_k^{(n-1)} = \sum_{j=k}^{k+n-1} [f_j(1+\delta_{k,j}^{(n-1)})] \prod_{i=k,\, i\neq j}^{k+n} \frac{z - tz_i}{z_j - z_i}\,(z_j - z_{k+n}), \tag{26}$$

$$(z - tz_k)\,\tilde{c}_{k+1}^{(n-1)} = \sum_{j=k+1}^{k+n} [f_j(1+\delta_{k+1,j}^{(n-1)})] \prod_{i=k,\, i\neq j}^{k+n} \frac{z - tz_i}{z_j - z_i}\,(z_j - z_k). \tag{27}$$

To simplify the notation let us define

$$1 + \phi_{k,j}^{(n)} = (1+\alpha_k^{(n)})(1+\delta_{k,j}^{(n-1)})$$

and

$$1 + \psi_{k,j}^{(n)} = (1+\beta_k^{(n)})(1+\delta_{k+1,j}^{(n-1)}),$$

bounded as follows

$$|\phi_{k,j}^{(n)}|,\, |\psi_{k,j}^{(n)}| \leqslant \epsilon_M d\,(1+(n-1)L^{n-2}) + \mathcal{O}(\epsilon_M^2). \tag{28}$$

Subtraction of (26) from (27) leads to

$$\tilde{c}_k^{(n)} = \sum_{j=k}^{k+n} f_j(1+\delta_{k,j}^{(n)}) \prod_{i=k,\, i\neq j}^{k+n} \frac{z - tz_i}{z_j - z_i},$$

where

$$1 + \delta_{k,j}^{(n)} = \frac{(1+\phi_{k,j}^{(n)})(z_j - z_{k+n}) + (1+\psi_{k,j}^{(n)})(z_k - z_j)}{z_k - z_{k+n}},$$

for $j = k+1, \ldots, k+n-1$, and

$$1 + \delta_{k,k}^{(n)} = 1 + \phi_{k,k}^{(n)}, \qquad 1 + \delta_{k,k+n}^{(n)} = 1 + \psi_{k,k+n}^{(n)}.$$

This leads to the estimation

$$|\delta_{k,j}^{(n)}| \leqslant \max\{|\phi_{k,j}^{(n)}|,\, |\psi_{k,j}^{(n)}|\}\, L, \qquad \text{for} \quad j = k, \ldots, k+n,$$

which, in combination with (28) and the fact that $L \geqslant 1$ (see (23)) results in $|\delta_{k,j}^{(n)}| \leqslant \epsilon_M nd\, L^{n-1} + \mathcal{O}(\epsilon_M^2)$, which completes the proof. ■

Although, in general, the constant L defined in (23) might be large, in the special cases the Algorithm I is numerically stable in sense (4). This happens, for example, for specially ordered knots. For details see the following proposition and remarks below it.

Proposition 1 *Assume that z_j, $f_j \in \mathbb{R}$ for $j = 0, 1, \ldots, N$ and $z, t \in \mathbb{R}$ are exactly representable in fl and* (20) *holds. If the knots z_j are monotonically ordered then the values $\tilde{p}_n(z;t) = \tilde{c}_0^{(n)}$ computed in fl by Algorithm I satisfy*

$$\tilde{p}_n(z;t) = \sum_{j=0}^{n} [f_j(1+\delta_j^{(n)})] \prod_{i=0,\, i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad |\delta_j^{(n)}| \leqslant \epsilon_M 5\, n + \mathcal{O}(\epsilon_M^2). \quad (29)$$

Proof. It is a direct consequence of Theorem 1 for $k = 0$. For monotonically ordered knots z_j the constant L in (23) is 1. ■

Remark. If $z_j \in \mathbb{C}$, $z_j = a + jh$, $j = 0, 1, \ldots, N$, $h = (b-a)/N$, $a, b \in \mathbb{C}$, then $L = 1$ and Algorithm I is also numerically stable, i.e. the condition (4) holds with the constant $k_N \approx (8 + 2\sqrt{2})N$.

In Section 5 we present numerical experiments showing that Algorithm I is not always numerically stable.

4.2 Error analysis of Algorithm II

Theorem 2 *Assume that z_j, $f_j \in \mathbb{C}$, $z \neq tz_j$ for $j = 0, 1, \ldots, N$ and $z, t \in \mathbb{C}$ are exactly representable in cfl and* (20) *holds. Then the values $\tilde{p}_n(z;t)$ computed in cfl by Algorithm II satisfy*

$$\tilde{p}_n(z;t) = \sum_{j=0}^{n} [f_j(1+\delta_j^{(n)})] \prod_{i=0,\, i\neq j}^{n} \frac{z - tz_i}{z_j - z_i}, \qquad |\delta_j^{(n)}| \leqslant \epsilon_M k_N + \mathcal{O}(\epsilon_M^2), \quad (30)$$

where

$$k_N = (N+1) \begin{cases} 5 & \textit{in the real case,} \\ 8 + 2\sqrt{2} & \textit{in the complex case.} \end{cases} \quad (31)$$

Proof. The proof is straightforward. Using (13)-(16) and (17)-(19) we obtain the following quantities, computed in fl or cfl

$$\tilde{b}_j^{(0)} = \frac{f_j}{z - tz_j}(1 + \Delta_j^{(0)}), \qquad j = 0, 1, \ldots, N,$$

and for $n = 1, 2, \ldots, N$

$$\tilde{A}_n = A_n(1 + \alpha_n),$$

$$\tilde{b}_j^{(n)} = \frac{\tilde{b}_j^{(n-1)}}{z_j - z_n}(1 + \beta_j^{(n)}), \qquad j = 0, 1, \ldots, n-1,$$

$$\tilde{b}_n^{(n)} = \frac{\tilde{b}_n^{(0)}}{\prod_{j=0}^{n-1}(z_n - z_j)}(1 + \gamma_n),$$

From (12) we obtain

$$\tilde{B}_n = \sum_{j=0}^{n} \tilde{b}_j^{(n)}(1 + \alpha_j^{(n)}), \qquad |\alpha_j^{(n)}| \leqslant \epsilon_M\, n + \mathcal{O}(\epsilon_M^2),$$

$$\tilde{p}_n(z;t) = \tilde{A}_n \tilde{B}_n (1 + \delta_n),$$

where

$$|\Delta_j^{(0)}|,\ |\beta_j^{(n)}| \leqslant \epsilon_M \left\{ \begin{array}{cl} 2 & \text{(real case)} \\ 5 + \sqrt{2} & \text{(complex case)} \end{array} \right. + \mathcal{O}(\epsilon_M^2),$$

$$|\alpha_n| \leqslant \epsilon_M \left\{ \begin{array}{cl} 2\,n + 1 & \text{(real case)} \\ (2 + \sqrt{2})\,n + 1 & \text{(complex case)} \end{array} \right. + \mathcal{O}(\epsilon_M^2),$$

$$|\gamma_n| \leqslant \epsilon_M \left\{ \begin{array}{cl} 2\,n & \text{(real case)} \\ (2 + \sqrt{2})\,n + 3 & \text{(complex case)} \end{array} \right. + \mathcal{O}(\epsilon_M^2),$$

$$|\delta_n| \leqslant \epsilon_M \left\{ \begin{array}{cl} 1 & \text{(real case)} \\ 1 + \sqrt{2} & \text{(complex case)} \end{array} \right. + \mathcal{O}(\epsilon_M^2).$$

Now the formulae (15), (16) yield for all $b_j^{(n)}$

$$\tilde{b}_j^{(n)} = b_j^{(n)}(1 + \phi_j^{(n)}), \tag{32}$$

where

$$|\phi_j^{(n)}| \leqslant \epsilon_M(n+1)\begin{cases} 2 & \text{(real case)} \\ 5+\sqrt{2} & \text{(complex case)} \end{cases} + \mathcal{O}(\epsilon_M^2) \tag{33}$$

and for $\tilde{B}_n$, $n = 1, 2 \ldots, N$

$$\tilde{B}_n = \sum_{j=0}^{n} \tilde{b}_j^{(n)}(1+\psi_j^{(n)}),$$

where

$$|\psi_j^{(n)}| \leqslant \epsilon_M(n+1)\begin{cases} 3 & \text{(real case)} \\ 6+\sqrt{2} & \text{(complex case)} \end{cases} + \mathcal{O}(\epsilon_M^2).$$

Finally, this and (32), (33) lead to (30), which proves the theorem. ■

Remark. In the case of divided differences algorithm (formula (3)) the constants are smaller, namely (Cf. [9])

$$|\delta_j^{(n)}| \leqslant \epsilon_M(n+1)\begin{cases} 3 & \text{(real case)} \\ 6+\sqrt{2} & \text{(complex case)} \end{cases} + \mathcal{O}(\epsilon_M^2).$$

5 Numerical experiments

To illustrate our results we present numerical experiments carried out in *MATLAB* with unit roundoff $\epsilon_M = 2.2e\text{–}16$.

We implemented all methods and performed many tests in order to investigate the behaviour of these methods, i.e. to compare the errors generated by each of them. This paragraph contains the results of some of these tests.

Algorithm II is general, two special cases are computing the Newton coefficients and computing the value of an interpolating polynomial. These are the cases we considered in our tests.

The first kind of tests involves computing the Newton coefficients. One type of the functions we interpolated was $f(z) = z^s$, $s = 2, 3, \ldots, 7$.

To compare our algorithm with the classical scheme of divided differences we evaluated the following quantities:

$$error1 = \frac{\max\limits_{n=s+2,\ldots,N} |\tilde{c}_n|}{\epsilon_M \max\limits_{n=0,1,\ldots,N} \sum\limits_{j=0}^{n} |f_j| \prod\limits_{i=0,\ i\neq j}^{n} \dfrac{1}{|z_j - z_i|}} \tag{34}$$

and

$$error2 = \frac{\max\limits_{n=s+2,\ldots,N} |\tilde{c}_n|}{\epsilon_M \max\limits_{n=0,1,\ldots,N} |c_n|}, \tag{35}$$

where $\tilde{c}_n$ and c_n are the computed and the exact Newton coefficients, respectively. Notice that if $f(z) = z^s$, then $c_{s+2} = \ldots = c_N = 0$ for $N \geqslant s+2$.

For numerically stable algorithms $error1$ is small, it is a measure of instability of an algorithm, which comes from the following chain of inequalities (see (4)-(7))

$$\max_{n=s+2,\ldots,N} |\tilde{c}_n| \leqslant \max_{n=0,\ldots,N} |\tilde{c}_n - c_n| \leqslant \epsilon_M k_N \max_{n=0,\ldots,N} \sum_{j=0}^{n} |f_j| \prod_{i=0,\ i\neq j}^{n} \frac{1}{|z_j - z_i|}. \tag{36}$$

Now it is easy to see that $error1$ should be less than or equal to the constant k_N in (4).

Tables 1 and 2 contain the respective values of $error1$ and $error2$ for Aitken's algorithm and Algorithm II applied to the function $f(z) = z^7$. The interpolation was based on 50 or 80 random knots ordered either monotonically or by Leja ordering. In all three tests problem was well-conditioned, with condition number (7) of order 1.

Table 1: The error (34) for the scheme of divided differences (Div. diff.) and Algorithm II for random knots, increasing (↑) and Leja orderings.

N	Div. diff. + ↑	Alg. II + ↑	Div. diff. + Leja	Alg. II + Leja
50	1.161255e+04	5.470269e−02	4.587882e−01	6.130860e−05
50	8.622460e+02	4.401986e−02	8.014843e−05	3.478342e−05
80	2.274724e+06	1.340639e−01	1.195999e−04	7.967939e−05

Table 2: The error (35) for the scheme of divided differences (Div. diff.) and Algorithm II for the same data as in Table 1.

N	Div. diff. + $\uparrow$	Alg. II + $\uparrow$	Div. diff. + Leja	Alg. II + Leja
50	2.019597e+04	9.513618e−02	4.587882e−01	6.130860e−05
50	9.928299e+02	5.068650e−02	8.014843e−05	3.478342e−05
80	2.278485e+06	1.342856e−01	1.195999e−04	7.967939e−05

Now let us turn our attention to evaluation of an interpolating polynomial.

Tests presented below were performed for the function $f(z) = z^7$. Interpolation was based on 100 equally spaced knots from the interval $[-1, 1]$, ordered either monotonically or by Leja ordering.

All graphs present the error (see (4)-(7) for the function $f(z) = z^7$)

$$error3 = \frac{|\tilde{p}_n(z,1) - z^7|}{\epsilon_M \max\limits_{n=0,1,\ldots,N} \sum\limits_{j=1}^{n} |f_j| \prod\limits_{i=1,\, i\neq j}^{n} \frac{|z - z_i|}{|z_j - z_i|}} \tag{37}$$

computed for equally spaced points from interval $[-98/99, 97/98]$, where $\tilde{p}_n(z,1)$ is the value of an interpolating polynomial $p_n(z,1)$ in (2) given either by Aitken's algorithm or by Algorithm II.

Figures 1 and 2 describe the results for Aitken's algorithm and Algorithm II, respectively, for monotonically ordered knots. They confirm theoretical results, that Aitken's algorithm is numerically stable for monotonically ordered points (increasingly or decreasingly), see Theorem 1.

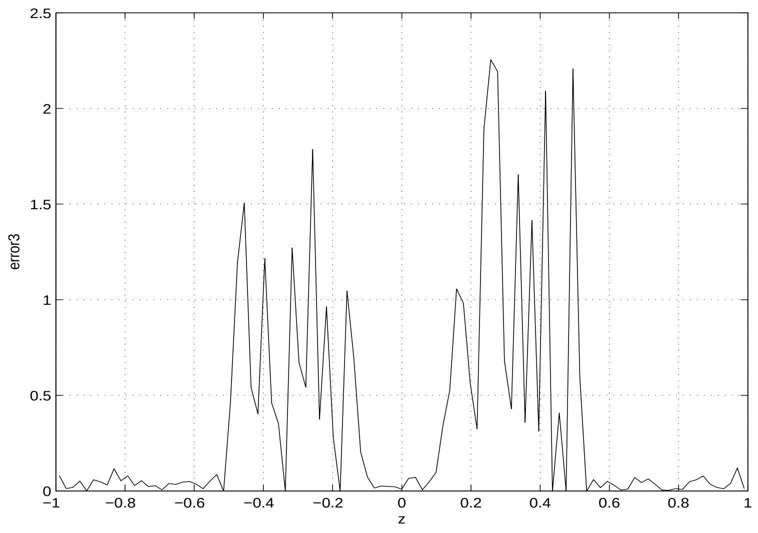

Figure 1: The error (37) for Aitken's algorithm for $f(z) = z^7$
with the knots being 100 equally spaced points
from the interval $[-98/99, 97/98]$, increasingly ordered.

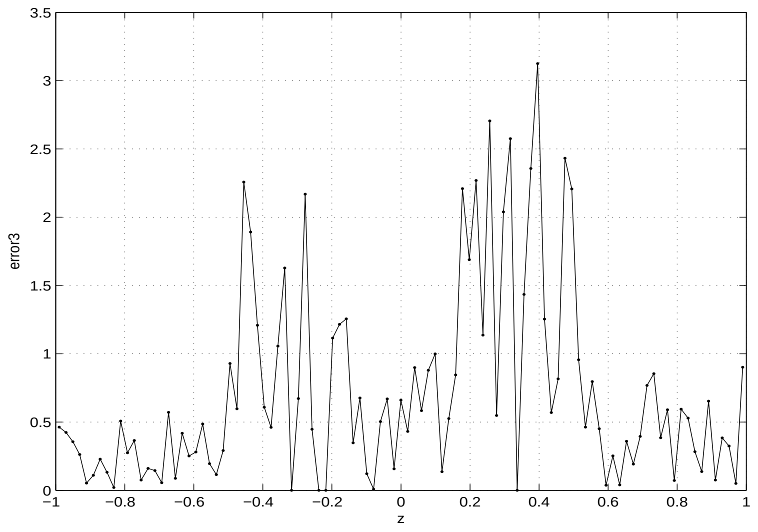

Figure 2: The error (37) for Algorithm II for $f(z) = z^7$
with the knots being 100 equally spaced points
from the interval $[-98/99, 97/98]$, increasingly ordered.

Figures 3 and 4 present the values of error (37) for Aitken's algorithm and Algorithm II, respectively, for knots ordered by Leja ordering.

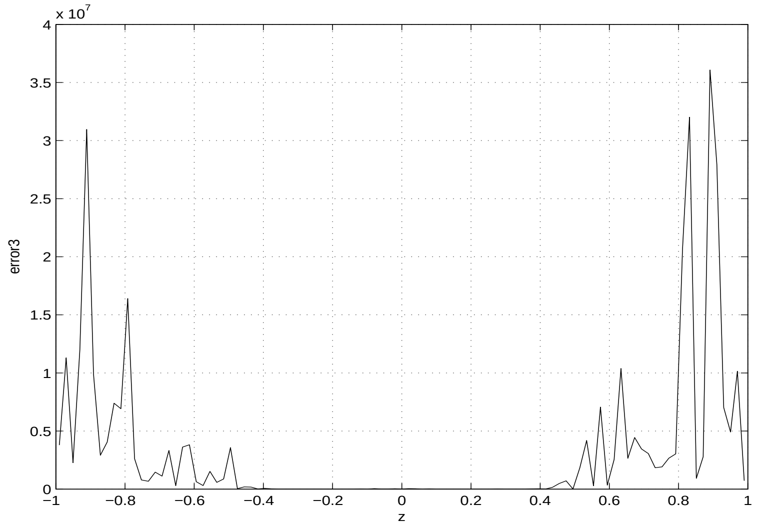

Figure 3: The error (37) for Aitken's algorithm for $f(z) = z^7$ with the knots being 100 equally spaced points from the interval $[-98/99, 97/98]$, ordered by Leja ordering.

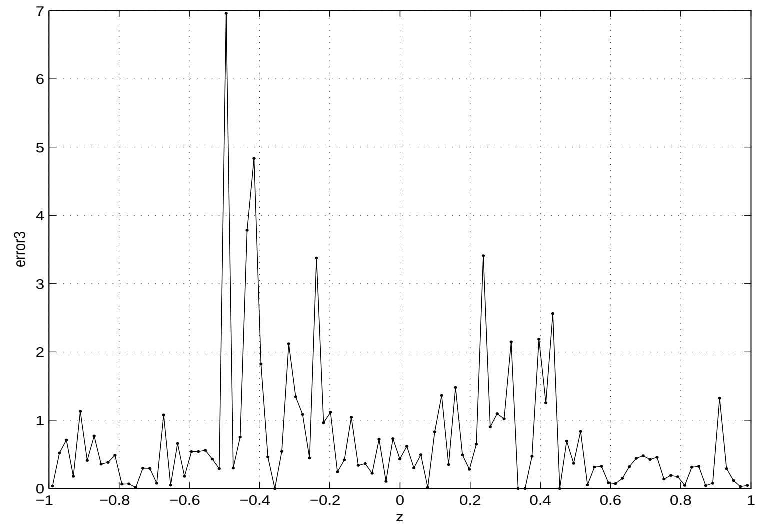

Figure 4: The error (37) for Algorithm II for $f(z) = z^7$ with the knots being 100 equally spaced points from the interval $[-98/99, 97/98]$, ordered by Leja ordering.

Table 3 contains values of $error3$ at five of the points in the data used to create figures 1-4.

Table 3: The error (37) for Aitken's algorithm and Algorithm II with knots ordered either increasingly or by Leja ordering.

z	Aitken + $\uparrow$	Alg. II + $\uparrow$	Aitken + Leja	Alg. II + Leja
$-.989899$	$8.078746e{-}02$	$4.623302e{-}01$	$3.791592e{+}06$	$3.593606e{-}02$
$-.791929$	$7.973497e{-}02$	$2.746146e{-}01$	$1.640714e{+}07$	$6.377801e{-}02$
$-.593960$	$3.579114e{-}02$	$2.806026e{-}01$	$6.421192e{+}05$	$5.397304e{-}01$
$-.395990$	$1.215628e{+}00$	$6.078142e{-}01$	$7.446058e{+}04$	$1.823442e{+}00$
$-.198021$	$2.787772e{-}01$	$1.115109e{+}00$	$3.299607e{+}03$	$1.115109e{+}00$

Note that Leja ordering, although in many cases helpful, is not a general remedy. Aitken's algorithm is more accurate for monotonically ordered knots than for Leja points. This is because of the constant L in Theorem 1. For monotone order L is always 1, for Leja ordering L may be big, for example in the test presented above $L = 197$.

References

[1] Å. Björck and V. Pereyra, Solution of Vandermonde systems of equations, Math. Comput. 24 (112) (1970) 893-903.

[2] D. Calvetti and L. Reichel, On the evaluation of polynomial coefficients, Numer. Algorithms 33 (2003) 153-161.

[3] G. Dahlquist and Å. Björck, Numerical methods, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

[4] J. i M. Jankowscy, Przegląd metod i algorytmów numerycznych, cz. 1, Warszawa, WNT (in Polish).

[5] W. Kahan and I. Farkas, Algorithm 167-Calculation of confluent divided differences, Comm. ACM 6 (1963) 164-165.

[6] A. McCurdy, K.C. Ng and B.N. Parlett, Accurate computation of divided differences of the exponential function, Math. Comput. 43 (168) (1984) 501-528.

[7] H. J. Rack and M. Reimer, The numerical stability of evaluation schemes for polynomials based on the Lagrange interpolation form, BIT 22 (1982) 101-107.

[8] L. Reichel, Newton interpolation at Leja points, BIT 30 (1990) 332-346.

[9] A. Smoktunowicz, Stability issues for special algebraic problems, Ph.D. Thesis, Univ. of Warsaw, 1981 (in Polish).

[10] J. Stoer and R. Bulirsch, Introduction to numerical analysis, Springer-Verlag, NJ, 1980.

[11] W. Werner, Polynomial interpolation: Lagrange versus Newton, Math. Comput. 43 (167) (1984) 205-217.

[12] J. H. Wilkinson, Rounding errors in algebraic processes, Notes on Applied Science No. 32, Her Majesty's Stationary Office, London, 1963.